U.S. NONPROVISIONAL PATENT APPLICATION

ACTIVE FILE SYSTEM

Patrick Brian Osterday
2631 Pancoast Ave
Cincinnati, OH 45211

Valiyolah Tadayon
5980 Winnetka Dr.
Cincinnati, OH 45236

James Edward Wilson
8175 Lyndhurst Ct.
Cincinnati, OH 45249

Charles Martin Zwick
6821 Simpson Avenue #6
Cincinnati, OH 45239

Attorney Docket No. 85252.484318

Ria Farrell Schalnat
Registration No. 47,058
FROST BROWN TODD LLC
2200 PNC Center
201 East Fifth Street
Cincinnati, Ohio 45202
(513) 651-6167

"Express Mail" mailing label number

EL583886655

8/31/01
Date of Deposit

I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR §1.10 on the date indicated above and is addressed to The Commissioner of Patents and Trademarks, Washington, D.C. 20231.

Tammi Carskadon
(Type or print name of person mailing paper of fee)

Tammi Carskadon
Signature

ACTIVE FILE SYSTEM

Patrick Brian Osterday
2631 Pancoast Ave
Cincinnati, OH 45211

Valiyolah Tadayon
5980 Winnetka Dr.
Cincinnati, OH 45236

James Edward Wilson
8175 Lyndhurst Ct.
Cincinnati, OH 45249

Charles Martin Zwick
6821 Simpson Avenue #6
Cincinnati, OH 45239

[0001] This application is a continuation-in-part application under CFR 1.53(b) which claims priority from the earlier filed non-provisional application Serial No. 09/421,808 which was filed on October 20, 1999, and which claims priority from the earlier filed provisional application Serial No. 60/156,676, which was filed on September 29, 1999.

TECHNICAL FIELD

[0002] The present invention relates generally to computerized file systems.

BACKGROUND OF THE INVENTION

[0003] A file system is a system that an operating system or program uses to organize and keep track of files. Files are collections of data or information, which are stored on a computer-readable medium and are identified by their filename. Almost all information stored in a computer must be in a file. There are many different types of files: text files, program files, directory files, and so on. Each type of file

stores different types of information. For example, program files store executable code; text files store text; directory files comprise information about a directory, etc. Known file systems organize files into a hierarchical structure by using directory files as containers for all other types of files.

[0004] The file system keeps track of files stored on a computer-readable medium/memory including hard disks. Hard disks utilize a file system to define the logical structures to control access to the storage on the hard disk. One of the most common PC file system types is the FAT (file allocation table) family, which includes FAT12/FAT16/VFAT/FAT32. The Windows NT and Windows 2000 operating systems use the New Technology File System (NTFS). Less common operating systems, such as OS/2, Linux (uses inodes) or BeOS, also utilize proprietary file systems.

[0005] In the FAT file systems, the file allocation table gives the FAT file system its structure. Data is stored in *clusters* of x-byte sectors on the hard disk (x generally equals 512, but the size of the byte sectors depends on how the hard drive is initially set up). The file allocation table maintains information about where the clusters are stored on the hard disk and whether or not the clusters are in use. Each cluster has an entry in the file allocation table. The cluster entries are chained together to accommodate larger files (one cluster entry contains a pointer to the next cluster entry until the end of file marker is found ... the clusters are not necessarily chained in consecutive order). Information about every file on the hard disk is stored in the folder/directory to which the file is assigned. The directory is a specially structured file that is interpreted in a particular way by the operating system. The directory comprises a table containing information/attributes about all the files and subdirectories depending from it. It also contains an identifier for

the first cluster of each file assigned to that directory. The operating system looks up the identifier of the first cluster in the file allocation table to determine the remaining clusters chained to it so that it may access the entire file. To find the information for a particular filename, the operating system searches for that filename in the directory identified in the path given by the user. If no path is given, the operating system will default to either the working directory or the root directory to look for the filename.

[0006] In the New Technology File System (NTFS), metadata files define the file system. Metadata files generally refer to something that is transcendent or self-referring. These metadata files basically contain data about data. The key metadata file is the Master File Table (MFT). The MFT is somewhat analogous to the file allocation table present in the FAT file systems. Information about files and directories in the MFT takes the form of attributes including the data attribute, which holds or points to the data contained in the file. Attributes may be contained entirely within the MFT or may be non-resident (located elsewhere on the hard disk and the MFT only contains a pointer to the alternate position.) NTFS provides certain system defined attributes but it also supports the creation of "programmer-defined" attributes. An application developer can create their own file attributes to govern the behavior of files and directories but such attributes are rather difficult to access. Furthermore it is not possible to change the fundamental nature of a file or a folder by modifying their attributes through this system.

[0007] The main difference between the FAT and NTFS file systems is that they both manage the hierarchical structuring of directories differently. In FAT, directories are responsible for storing most of the key information about files and the files themselves only contain

data. In NTFS, files are collections of attributes, so they contain their own descriptive information, as well as their own data. So in NTFS, a directory stores information about the directory but not about the files contained in that directory.

[0008] The structure and operation of other types of file systems are within the knowledge of those skilled in the art of the operating systems to which said file systems belong.

[0009] Known file systems utilize a hierarchical directory structure to organize files on the hard disk. File systems utilize a logical tree format with a root/trunk (root directory), branches (directories), twigs (subdirectories), and finally leaves (files). Branches depend from the root. Twigs depend from the branches and from each other. Leaves are the final extension of the structure. In other words, neither branches, twigs, nor additional leaves may depend from a leaf.

[0010] Operating systems provide their own file management system; however, it is possible to interface a separate file management system with the operating system's file management system. These systems, such as the Network File System (NFS), interact smoothly with the operating system but provide more features, such as improved backup procedures, stricter file protection, and remote access via a network.

[0011] NFS is a client/server application that lets a computer user view and optionally store and update files on a remote computer as though they were on the user's own computer by sending files/updates back and forth with Transmission Control Protocol/Internet Protocol (TCP/IP) or User Datagram Protocol (UDP). Using NFS, all or a portion of a remote file system is mounted on the server across homogenous and heterogeneous systems. The portion of your file

system that is mounted (designated as accessible) can be accessed with whatever privileges go with your access to each file (read-only or read-write). WEBNFS, which is implemented by Sun Microsystems, allows accessing by NFS users through Web browsers.

[0012] CIFS defines a standard remote file system access protocol for use over the Internet, which enables groups of users to work together and share documents across the Internet or within their own corporate intranets. CIFS enables collaboration on the Internet by defining a remote file access protocol that is compatible with how applications already share data on local disks and network file servers.

[0013] The development of a supplemental file management system to support the native file system/ underlying file system of a given operating system is within the abilities of one skilled in the art. The invention of this application relates to a file management system capable of interfacing with the native operating system's file system that includes specific functionality heretofore not known in the art.

SUMMARY OF THE INVENTION

[0014] The above-described file systems, while providing a standard model for organizing files in a stand-alone file system and organizing/sharing files in distributed file systems, follow the same hierarchical tree structure (root-branch-twig-leaf) that has been used since the 1950s. Furthermore, the ability to extend the attributes of the file system has been under-utilized both in terms of the kinds of intelligence added to the file system as well as providing an interface which allows non-programmer users to extend these attributes.

[0015] The invention comprises a program for supporting a both conventional and distributed file systems (including Internet-based) in the following manner:

[0016] A graphical user interface (GUI) and an application-programming interface (API) are provided to allow users to extend the attributes of files and directories in the file system. The GUI further presents the user with the ability to view and virtually organize files/objects, and directories far differently than the traditional hierarchical tree structure. Objects include any type of file excluding directory and subdirectory files.

[0017] The GUI modifies the presentation of the standard hierarchical tree structure to virtually allow objects (leaves) to act as containers for other objects or directories/folders. In conventional file systems, a "leaf" cannot be a parent to a branch, twig or other leaf. An object cannot be the parent of other objects, subdirectories, or directories. Extending the conventional file system to allow users to associate files, subdirectories, and directories with a parent file/object provides an entirely new organizational model. Files may comprise document files (Microsoft Word, Word Perfect), picture files (GIF, TIFF, JPEG), tables/spreadsheets (Excel), calendars, program files (executable code), movies (MPEG), music (MP3), contacts, uniform resource locators (URLs), bulletin board postings, contacts, and any other kind of information that may be stored for future use. In one instance, a department report may have a directory depending from it, which contains individual files comprising contact information for each member of the department. The department report may further have files comprising specific URLs for accessing websites relevant to the report on the Internet. Finally, the report may have files comprising a bulletin board for discussing the department report depending from the report.

[0018] Intelligence is added to the file system by allowing a user to define rules to be applied to the objects and directories present in the file system. The rules are defined/stored in the metadata, which is stored in an associated database, which may also describe/define the virtual file/object/directory structure. A scripting language may also be provided, as part of the API, to allow programmers to access the functionality of the system more powerfully.

[0019] In one embodiment the invention comprises an enhanced computerized file system comprising an underlying computerized file system; a database wherein said database provides storage for at least one pointer corresponding to a location associated with an object in said underlying file system, and at least one instruction corresponding to a virtual location associated with said object; a graphical user interface wherein said graphical user interface further provides a manipulable display which presents said virtual location of said object, and a wizard which presents at least one screen comprising at least one step for defining a rule for association with said object.

[0020] In another embodiment the invention comprises an enhanced computerized file system comprising an underlying computerized file system; a database wherein said database provides storage for at least one pointer corresponding to a location associated with a plurality of objects in said underlying file system, comprising at least a first object and a second object, wherein said plurality of objects comprise one or more of the following: a text file, music file, multimedia file, compressed file, uniform resource locator, contact, memo, bulletin board posting, or calendar; and at least one instruction corresponding to a virtual location associated with each of said objects; a graphical user interface which provides a

manipulable display which presents said virtual location of said objects and wherein said second object may be assigned as a virtual child of said first object.

[0021] In another embodiment the invention comprises an enhanced computerized file system comprising an underlying computerized file system; a database wherein said database provides storage for at least one pointer corresponding to a location associated with a plurality of objects in said underlying file system, comprising at least a first object and a second object; and at least one instruction corresponding to a virtual location associated with each of said objects; and wherein said database also provides storage for at least one pointer corresponding to a location associated with a directory in said underlying file system; a graphical user interface comprising a manipulable display which presents said virtual location of each of said objects wherein said second object may further virtually comprise a child of said first object, and a wizard comprising at least one screen comprising at least one step for defining a rule for association with each of said objects or said directory.

[0022] In another embodiment an enhanced computerized file system comprising an underlying computerized file system; a database including a portion of memory for storing at least one pointer corresponding to a location associated with a plurality of objects in said underlying file system, comprising a first object and a second object; a portion of memory for storing at least one instruction corresponding to a virtual location associated with each of said objects; and a portion of memory for storing at least one pointer corresponding to a location associated with a directory in said underlying file system; a graphical user interface (GUI) comprising

a manipulable display which presents said virtual location of each of said objects wherein said second object may further comprise a child of said first object, and a wizard which presents at least one screen comprising at least one step for defining a rule for association with each of said objects or said directory; and an application programming interface (API) wherein said API: accepts commands from a user from said GUI (user commands); translates said user commands into a set of native commands to be run against said database and against said underlying file system to obtain an output; processes said output; and displays said output on said GUI.

[0023] In another embodiment the invention comprises a method of enhancing a computerized file system comprising associating an underlying computerized file system of a computer system with a database; providing a memory location in said database for a pointer corresponding to a location for a plurality of objects stored in said underlying computerized file system, comprising at least a first object and a second object, wherein said plurality of objects comprise one or more of the following: a text file, music file, multimedia file, compressed file, uniform resource locator, contact, memo, bulletin board posting, or calendar; providing a storage location for an instruction corresponding to a virtual location associated with each of said plurality of objects; providing a storage location for a rule corresponding to each of said plurality of objects in said database; providing a graphical user interface (GUI); presenting a manipulable display of said virtual location of each of said plurality of objects wherein said second object may further comprise a child of said first object, and providing a wizard which presents at least one screen comprising at least one step for defining said rule for association with each of said objects or a directory associated with said underlying file system; and providing an

application programming interface (API) wherein said API accepts commands from a user from said GUI (user commands); translating said user commands into a set of native commands to be run against said database and against said underlying file system to obtain an output; processing said output; and displaying said output through said GUI.

[0024] Additional advantages and other novel features of the invention will be set forth in part in the description that follows and in part will become apparent to those skilled in the art upon examination of the following or may be learned with the practice of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0025] The accompanying drawings incorporated in and forming a part of the specification illustrate several aspects of the present invention, and together with the description and claims serve to explain the principles of the invention. In the drawings:

[0026] **Figure 1** is a representation of the virtual graphical user interface for one embodiment of the invention.

[0027] **Figure 2** is a screenshot of a wizard employed by one embodiment of the invention which allows a user to define intelligence (a rule) for an object type by defining an event/action to be applied to an object type.

[0028] **Figure 3** is a screenshot of a wizard employed by one embodiment of the invention which allows a user to define a condition to be applied to an object type.

[0029] **Figure 4** is a screenshot of a wizard employed by one embodiment of the invention which allows a user to further define a condition to be applied to an object type.

[0030] **Figure 5** is a screen shot of a wizard employed by one embodiment of the invention which allows a user to define the recipients of an Automatic Notification and continue the application of intelligence to a particular object.

[0031] **Figure 6** is a screen shot of a wizard employed by one embodiment of the invention which allows a user to provide a name and description for a rule designed with the system.

[0032] **Figure 7** is a representation of the virtual graphical user interface for one embodiment of the invention which includes the association of a rule to a folder.

[0033] **Figure 8** is a representation of the structure of the system.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0034] Reference will now be made in detail to the various embodiments of the invention, examples of which are illustrated in the accompanying drawings, wherein like numerals indicate the same elements throughout the views.

[0035] To better describe the present invention, a definition of certain terms is provided immediately below:

[0036] Application Programming Interface—An application program interface (API) comprises instructions encoded on a computer readable medium for the specific method prescribed by a computer operating system or by an application program by which a programmer writing an application program can make requests of the operating system or another application.

[0037] Database—a large collection of data organized in a manner to facilitate quick retrieval.

[0038] Database (Active)—a database that includes a management system that can automatically react to events, such as database transactions, time events, and external signals that trigger the evaluation of a condition; and if the condition evaluates to be true, a specific action is carried out. Generally, an active database consists of a passive database and a set of active rules.

[0039] Database (Traditional/Passive)—a database system that is passive in behavior, which either periodically polls the database, or embeds or encodes event detection and execution of related action in the application code.

[0040] Field—an area of a database record into which a particular item of data is stored.

[0041] File—a block of information that is stored on a computerized electronic storage device; in typical installations, a file has an associated "filename."

[0042] File System—in a computer, a way in which files are named and where they are placed logically for storage and retrieval. The DOS, Windows, OS/2, Macintosh, and UNIX-based operating systems all have file systems in which files are placed somewhere in a hierarchical (tree) structure. A file is placed in a directory (*folder* in Windows) or subdirectory at the desired placed in the tree structure. The term also encompasses the part of an operating system or program that supports a file system.

[0043] Folder—an organizational unit that houses objects (including files) and other folders - sometimes also referred to as a directory in a disk operating system. In the Windows, Macintosh, and some other

operating systems, a folder is a named collection of related files that can be retrieved, moved, and otherwise manipulated as one entity. The folder has a data structure that represents the storage location it comprises in the file system.

[0044] Graphical User Interface (GUI) —A GUI is a graphical (rather than purely textual) user interface to a computer. Applications typically use the elements of the GUI that come with the operating system and add their own graphical user interface elements and ideas. Elements of a GUI include such things as windows, pull-down menus, buttons, scroll bars, iconic images, wizards, the mouse, and more. With the increasing use of multimedia as part of the GUI, sound, voice, motion video, and virtual reality interfaces seem likely to become part of the GUI for many applications. A system's graphical user interface along with its input devices is sometimes referred to as its "look-and-feel." When creating an application, many object-oriented tools exist that facilitate writing a graphical user interface. Each GUI element is defined as a class widget from which you can create object instances for your application. You can code or modify prepackaged methods that an object will use to respond to user stimuli.

[0045] Internet—a set of networks interconnected with routers.

[0046] Module—a segment of code written to certain specifications to perform a specific function, which allows for the creation, modification, or deletion of a specific object that the module was created for.

[0047] Network—an interconnected group of computers that connect to and cooperate with each other.

[0048] Object—a single entity of a specific type of information. As an example, a file is an object, and an event is an object. Calendars,

contacts, URLs, and other embodiments of information may also be considered to be objects.

[0049] Operating System—software that controls the operation of a computer, including the organization of its memory components, such as a hard disk drive.

[0050] Packet—a unit of data sent across a network.

[0051] Physical Storage—a medium onto which data can be entered, in which it can be contained (i.e., stored) and from which it can be retrieved at a later time; examples include a hard disk, semiconductor memory (e.g., RAM, ROM, EPROM, EEPROM, NVRAM), compact disc, etc.

[0052] Pointer—In programming, a pointer is a special type of variable that holds a memory address (that is, it *points* to a memory location) in a computer-readable storage medium.

[0053] Protocol—In information technology, a protocol is the special set of rules that end points in a computer connection use when they communicate. Protocols exist at several levels in a computer connection. There are hardware protocols and protocols between each of several functional layers at the software level. Both end points must recognize and observe a protocol. Protocols are often described in an industry or international standard.

[0054] Record—an ordered set of fields; a row in a database table.

[0055] Router—a device that forwards packets between networks.

[0056] Server—In general, a server is a computer program, encoded on a computer readable medium including a computer's hard drive, that provides services to other computer programs in the same or other computers. The computer that a server program runs in is also frequently referred to as a server (though it may contain a number

of server and client programs). In the client/server programming model, a server is a program that awaits and fulfills requests from client programs in the same or other computers. A given application in a computer may function as a *client* with requests for services from other programs and also as a *server* of requests from other programs. Specific to the Web, a Web server is the computer program (encoded on a computer readable medium) that serves requested HTML pages or files. A Web *client* is the requesting program associated with the user.

[0057] Table—a collection of records in a database.

[0058] User Interface a computer interface which includes GUIs, text-and-keyboard oriented command line interfaces, and non-graphical *menu-based interfaces*, which let you interact by using a mouse rather than by having to type in keyboard commands.

[0059] Virtual—the quality of effecting something without actually being that something; i.e., a virtual file structure does not have to mirror the corresponding physical layout of files/directories and pointers of files/directories on a physical storage device physical. Typically, a normal operating system's low level file system does mirror the physical layout therefore a low level file system would not be a virtual file system. A virtual child to virtual parent relationship, in a virtual file system, reflects an organizational dependency that may not be reflected in the physical layout of those objects in the structure of the physical storage device.

[0060] Web Browser—a program that allows a person to read hypertext.

[0061] Wizard—A utility within an application that helps a user use the application to perform a particular task. For example, a "letter wizard" within a word processing application would lead the user through the steps of producing different types of correspondence.

[0062] To achieve the foregoing and other advantages, and in accordance with one aspect of the present invention, an improved computerized file system (*active file system*) is provided that is accessible by users on a stand-alone computer, over the Internet, or via a local area network (LAN) or wide area network (WAN). The *active file system* provides intelligence to communicate and manage data indexed by the file system.

[0063] Internet users typically operate personal computer platforms having a web browser software program that communicates with an Internet Service Provider (ISP). The ISP launches the users onto the Internet itself. On the other end, a web server configured by the *active file system* of the present invention receives messages from the users, and transmits messages back to those users as they "logon" to the server.

[0064] The invention comprises a graphical user interface (GUI) and an application-programming interface (API). Configuring rules, groups, etc. within the file system may be wizard-based to simplify configuration, setup and on-going management.

Extension of Hierarchical Tree Structure

[0065] Referring to Figure 1, the GUI (10) represents files/objects (leaves) (110) as containers for other files or directories/folders (120). This virtual structure is not physical in nature like a normal operating system's low level file system because a virtual structure does not have to exist on a physical storage device (hard drive, etc.). Rather, in one embodiment, the structure may exist as one or more tables in a database. This database performs a similar role to the FAT of the FAT family or the MFT of the NTFS; however, the functionality enabled by the database goes beyond what FAT and MFT provide. The GUI (10) may also provide a properties window to illustrate

various properties (author, date created, file permissions, applicable rules, etc.) of a selected object within the virtual structure.

[0066] Referring to Figure 8, when a file/object is uploaded into the system by a client/user (800), the file/object is physically stored on a receiving server (810), which comprises a database (820) and an underlying file system (840). The underlying file system may comprise the native file system of the computer system's operating system or it may comprise an intermediary file system (such as WebNFS) which interacts with the native file system. Regardless of where the file/object is actually stored on the server, the appearance of the file's/object's location (virtual location) via the GUI (10) is based on a setting in the database (820). This database (820) also contains a pointer to the location of the file in the underlying file system (840). This system provides the further advantage of appearing to store multiple files with the same "display name" in the file system because a unique file identifier may be generated for each file uploaded to the system to distinguish its storage location in the underlying file system.

[0067] The *active file system* keeps its metadata (information about folders and objects) (830) in conventional databases (820). Incorporating metadata (830) into the database (820) allows the addition of new information about the folders (120) and objects (110, 120, 130) without modifying the file system directly.

Application of Rules to Files/Folders

[0068] Second, intelligence is added to the file system by allowing a user to define rules/workflow (830) that may be attached to the files and directories of the underlying file system (840) through metadata which describes/defines these rules (830).

[0069] Referring to Figures 2-6, rules allow users of the *active file system* to build logic to automatically perform specific actions through the GUI (20). The logic is performed/executed by a logic layer in the system (810). This logic is triggered based on an initial triggering event (210). The triggering event (210) may be related to a single object (file/directory) (200) in the file system, multiple objects, or may be a date/time-sensitive trigger. Responsive actions (220) include features such as sending notifications, requesting approvals, and executing workflows. An important feature of the *active file system* is its active behavior. Rules can be added to the files and directories of the *active file system* to monitor certain events and respond to them. The active aspects of the present invention concentrate on the file system (840) to both monitor and control certain actions by users, and then perform a predetermined task upon the occurrence of such predetermined events (210).

[0070] Triggering events (210) may comprise simple file operations (40) or "complex events" comprising a sequence of "simple events" (50). These complex events (as triggering events) are completely programmable, and their resulting actions can be similarly "complex" from the standpoint that they are also completely programmable, and further are able to launch completely separate application programs.

[0071] The basic logic for a rule equals "When {Trigger} If {Condition} Then Do {Action}" (210, 30-40, 440). Triggers (210) include upload, download, create, delete, rename, access, view, edit, save, move from, move to, copy to, copy from, and approval received. Actions (440) include simple actions, conditional actions, nested conditional actions, CASE actions, sending a notification (via e-mail, fax, pager, phone), requesting an approval, executing an action, executing a workflow, setting an object property, changing

rights, building a date/time trigger, running an application (including customer applications, word processing, accounting, engineering, statistical, graphical, CAD, compression, encryption, etc.), and/or building a visibility constraint. All of the rules specified may be time sensitive, recurring, or immediate. All of this information preferably is stored in a database, in which all of the defining triggering events and triggering actions are preferably held in a single large table of the database. Furthermore, referring to Figure 5, actions may be directed to particular users or groups of users (500).

[0072] Referring to Figure 7, the GUI (10) may represent the constructed rule (150) in the enhanced tree structure. In Figure 7, a rule (150) is applied to a folder (140). Individual components of a rule (150) may also be shown as sub-elements (160-180).

Automatic Notification Example

[0073] The automatic notification function has the capability to send messages automatically to other users that have an interest in a particular file/directory that is the subject of a predefined rule. There is complete flexibility as to just what events can be specified as becoming a "triggering event," (210) and there is complete flexibility as to "who" receives an automatic notification message (500). The automatic notification messages would normally be sent to users who are grouped in some logical (or at least predetermined) manner. A function may be provided to group users into such logical categories (510).

[0074] Automatic notifications are based on user, group, time or any other conditional attribute. Access to folders and objects can also be easily configured based on individual rights, times, or other conditions. In another embodiment, the *active file system* can

coordinate connectivity to email, phones, pagers, faxes, and other means of communication through the use of a messaging server and VoIP technologies. For example, when a file is added, modified or deleted, specified users or groups are automatically notified.

[0075] - When file created, e-mail Susan.

[0076] - When file uploaded, if file.size > 1 Meg, email Jim.

[0077] - When file uploaded, email the "everyone" group.

[0078] - When file uploaded, e-mail zwick + 14 days

[0079] - When DATE = 2/19/02, e-mail Bob.

[0080] -When file uploaded, if DATE = 5/20/02, e-mail Joe.

[0081] Attributes can be designed which provide notification to users or groups of users when certain triggering events occur. In conventional systems, if the first user modifies a particular file, only that first user is aware of this modification. A second user who may also have access to that same file would not be made aware of any such modifications, at least not without knowing the time and date stamps of previous saves of that same file. Using rules, an administrator in New York may create a folder on an *active file system* server. The administrator may define a rule to notify a group defined by the administrator to include User 1, User 2, and the administrator via their pagers if a file is uploaded or modified within the folder. The administrator may create/upload a file to that folder stored on the centralized *active file system* server. User 1 and User 2, upon receiving their notifications, may log onto the server and access the recently uploaded file. User 2 may modify the document. This will again trigger the administrator-defined rule and cause a notification message to be sent to the administrator and

User 1. Thereafter, User 1 and the administrator may log onto the *active file system* server to view the modified document.

[0082] In one embodiment, automatic notifications are transmitted via E-mail technology. Automatic notification messages, however, are completely configurable and may encompass both asynchronous and synchronous messaging as well as the launching of other programs or combinations thereof. Automatic notification signals users to come to the file rather than sending the file to each specified user (thereby decreasing network load). Finally, automatic notifications may contain links to the *active file system*, and the link will open the user's web browser and take the user directly to the folder or object that was just triggered.

Workflow Example

[0083] A user can program the *active file system* to set up a workflow automation process that allows the user to take a process from beginning to end by notifying the proper users when items are completed, if they are behind schedule, what action is required on their behalf, etc.

[0084] - When file.upload, folder.create ("chucky").

[0085] - When file.upload, execute the "process file" workflow.

[0086] - When file.create, set file.author = "zwick".

[0087] An example of workflow automation using an *active file system* is illustrated in the following: X is a small business, which outsources its payroll processing to Y. On a bi-weekly basis, an employee from X needs to open and edit an excel spreadsheet, update the working hours for each employee, enter any sick time or vacation time taken, and have the spreadsheet compute the weekly pay, sick time and vacation time balances. Y then needs to feed the data from

the spreadsheet into their payroll-processing system which prints and sends X's paychecks. The *active file system* may be used in the following manner to facilitate this process: an employee from Y gives an X employee rights to an Excel spreadsheet in Y's directory on a central file server. The employee from X edits the file, entering each employee's working, sick and vacation hours for the current pay period. Then the X employee saves the file to the central file server. This triggers a rule to notify an employee at Y that the spreadsheet has been updated and is ready for further processing. The employee at X may also associate additional files with the spreadsheet; for instance, the X employee may create an additional file containing questions regarding discrepancies in the spreadsheet. A rule may be set to notify the employee at Y when such a file is created and saved to the central file server. For the applicable pay period, another rule may be set for the spreadsheet that is triggered when the date equals the fourteenth (14^{th}) of the month. This rule will send the spreadsheet to a program and execute said program to processes the data and print out the paychecks. Finally, another rule may be set on the spreadsheet to move the file to an archive after a certain number of days has passed after the mailing of the paychecks. The *active file system* streamlines and automates a process that heretofore would have entailed redundant communications, multiple (and possibly conflicting) copies of the spreadsheet, and significantly more input from the human counterparts involved.

[0088] In another embodiment, the invention may be utilized to effect the timed viewing of a document/file. A user may program a rule to control the consumption of content based on time or bytes allotted. For instance, if watching a video costs ten cents/minute and the viewer has allocated $1, the rule should cut off access to the document/file after ten minutes.

[0089] In another embodiment, a user may program a rule to enforce type checking for a particular container so that a word processing document is not placed in a container programmed to receive only JPEG files.

[0090] In another embodiment, a visibility constraint may be implemented. A visibility constraint would limit access to a "games" folder to certain times of day to avoid play during normal work hours and an automatic notification could be set to inform a manager when employees are trying to access the games folder during normal working hours. In another embodiment the virtual file system could "hide" the games folder from normal users during the prohibited periods as opposed to merely denying access to the folder. The same principal could be used to tie the visibility of certain files/objects/folders to the logon ID of the user.

Application Programming Interface (API)

[0091] Referring to Figure 8, the API (810) comprises the logic, which executes the rules (830) embodied in the database (820). These rules affect the files/directories of the underlying file system (840). In one embodiment the API operates as follows: a user requests an object (800) via HTTP, WebDAV, or other protocols; this request is received by a server comprising a logic layer (810), operating system, database (820), and underlying file system (840); the logic layer (810) translates the user's request into a format that may be used to access a pointer to the location of the file in the underlying file system (840) through the database (820); as the logic layer (810) accesses the file's pointer, it may also scan the database (820) for any rules/workflow defining that file/directory; finally, the

object is sent back to the client through the logic/protocol layer (810).

[0092] In another embodiment, the API of the invention may be exposed to allow skilled users to directly manipulate/extend the attributes of files/directories. Exposure of the API, through a scripting language, would provide unlimited flexibility for a programmer to create new rules for any situation that doesn't currently exist within the *active file system* wizard's rules. Furthermore, any existing rule could be altered with this scripting language.

[0093] In original equipment manufacturer environments, the *active file system* can operate in the background as the foundation of a customized solution. Original Equipment Manufacturers can add their own modules to make the *active file system* more extensible.

[0094] In one embodiment, the application programming interface (API) allows users to enter commands to the system through the GUI that are then translated into a format that can be understood by the system and the underlying file system. This embodiment allows for switching out the Database Software (e.g., Oracle, DB2, SQL Server, etc.), Web Server Software (e.g., Internet Information Server – IIS, Cold Fusion, Java Servlets, SQL Stored Procedures, O'Reilly's Website, Apache, etc.), Operating System Software (e.g., Windows NT, Linux, Solaris, ColdFusion, etc.), or Middleware Software Components (e.g., ASP, PHP, etc.). The *active file system* network appliance would comprise the *active file system* server including the necessary hardware and software. Software would include a secure http server running on a fast, scalable operating system, with a back-end database, in addition to the *active file system* software running on the middleware component. In a web-based embodiment of the invention, the *active file system* preferably uses open standards (Java, html, and xml) that allow universal

access to a server employing such a file system. Therefore, the invention can, in one embodiment, accept CIFS or NFS client requests and process these requests transparently.

[0095] Still other advantages and embodiments of the present invention will become apparent to those skilled in this art from the following description and drawings wherein there is described and shown several embodiments contemplated for carrying out the invention. As will be realized, the invention is capable of other different embodiments, and its several details are capable of modification in various, obvious aspects all without departing from the invention. Accordingly, the drawings and descriptions will be regarded as illustrative in nature and not as restrictive. It is intended that the scope of the invention be defined by the claims appended hereto.